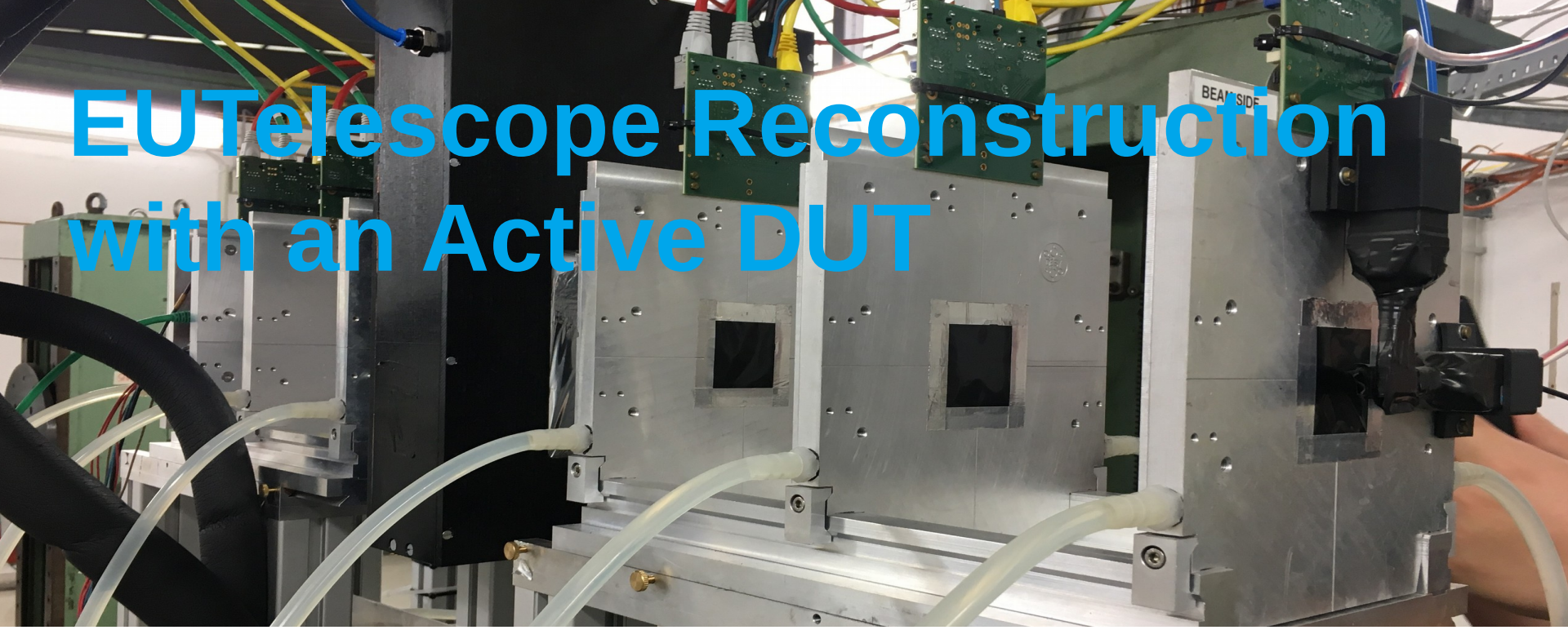


EUTelescope Reconstruction with an Active DUT

Edoardo Rossi, Jan-Hendrik Arling, Cyril Becot,
Jan Dreyling-Eschweiler, Michaela Queitsch-Maitland

7th Beam Telescopes and Test Beams Workshop, CERN, 15/01/2019

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

EUTelescope

DESY

Running EUTelescope

- For local Linux installation

  http://eutelescope.web.cern.ch/content/installation

- Instructions for docker image on lxplus or local here
- Instructions for installation on lxplus here (it takes >> 1 hour)
- A working installation is provided on lxplus and can be used if you have access to afs and cvmfs. To use it:

  1) scp -r /afs/cern.ch/work/e/edrossi/public/BTTBExample/v01-19-02/Eutelescope/master/jobsub/examples /path/you/want

  2) Enter into the folder GBL_DUT and open config.cfg. Modify BasePath to the path to your example folder

  3) source /afs/cern.ch/work/e/edrossi/public/BTTBExample/v01-19-02/Eutelescope/master/build_env.sh

  4) source /afs/cern.ch/work/e/edrossi/public/BTTBExample/v01-19-02/init_ilcsoft.sh

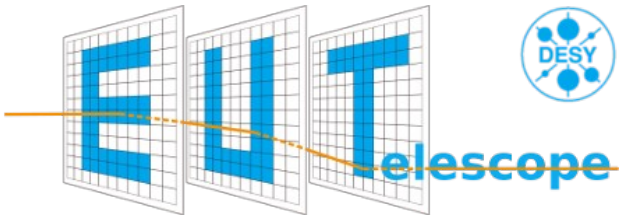
Update EUTelescope for the Tutorial

If you have installed the master branch, you change EUTelescope version to the one used in this tutorial. It can be done with git:

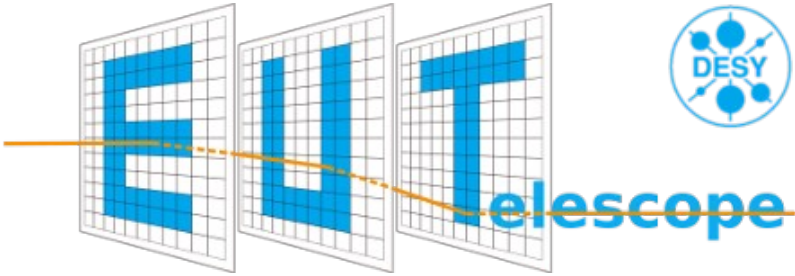
- cd $ILCSOFT/v01-19-02/Eutelescope/master/
- source ../../init_ilcsoft.sh
- source build_env.sh
- git checkout BTTB7
- cd build
- rm -rf * && cmake ..
- make install

About this Tutorial

- The aim is to show an example of reconstruction of data with an active DUT (and a reference plane) using EUTelescope and GBL
- Interrupt me at any time if you have a question!
- There is a lot of work on-going on EUTelescope. If you have questions, need help or have ideas for a new functionalities, get in touch with us!



What is EUTelescope?



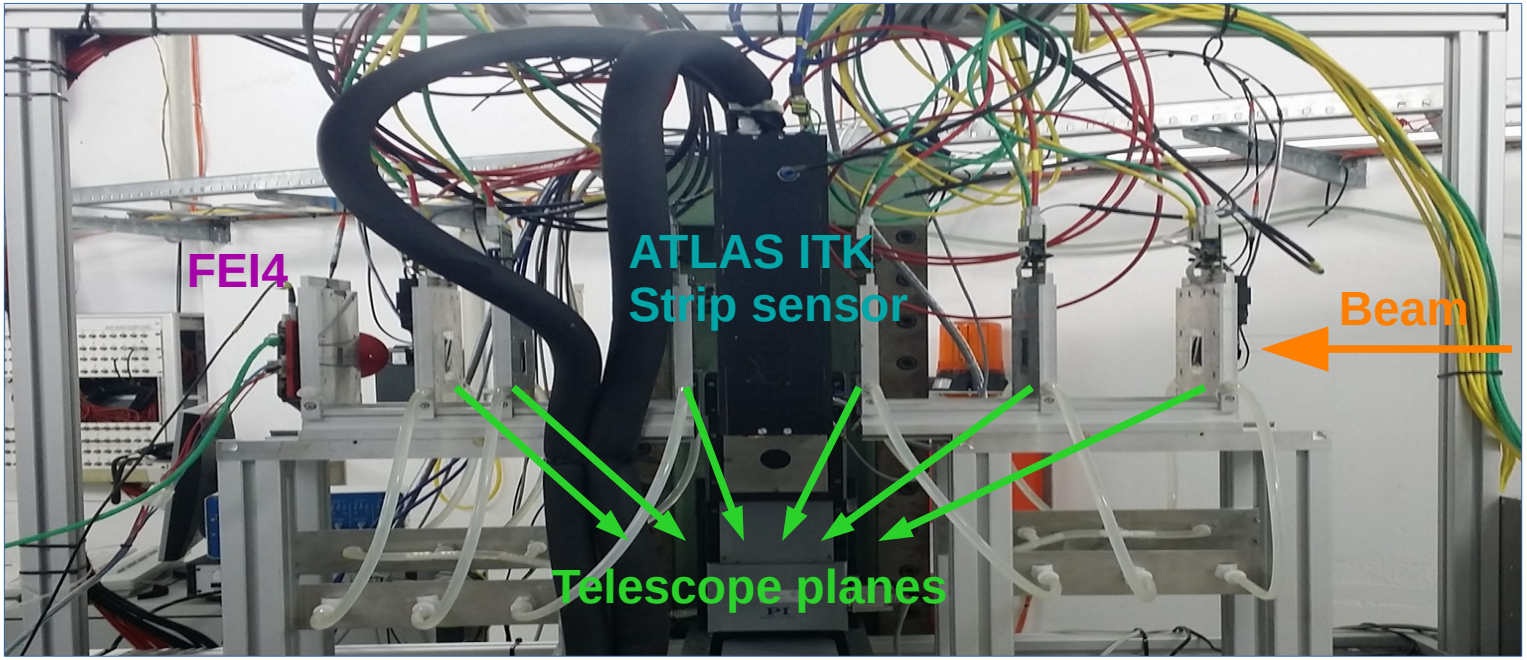
- EUTelescope can be found here:

  https://github.com/eutelescope/eutelescope

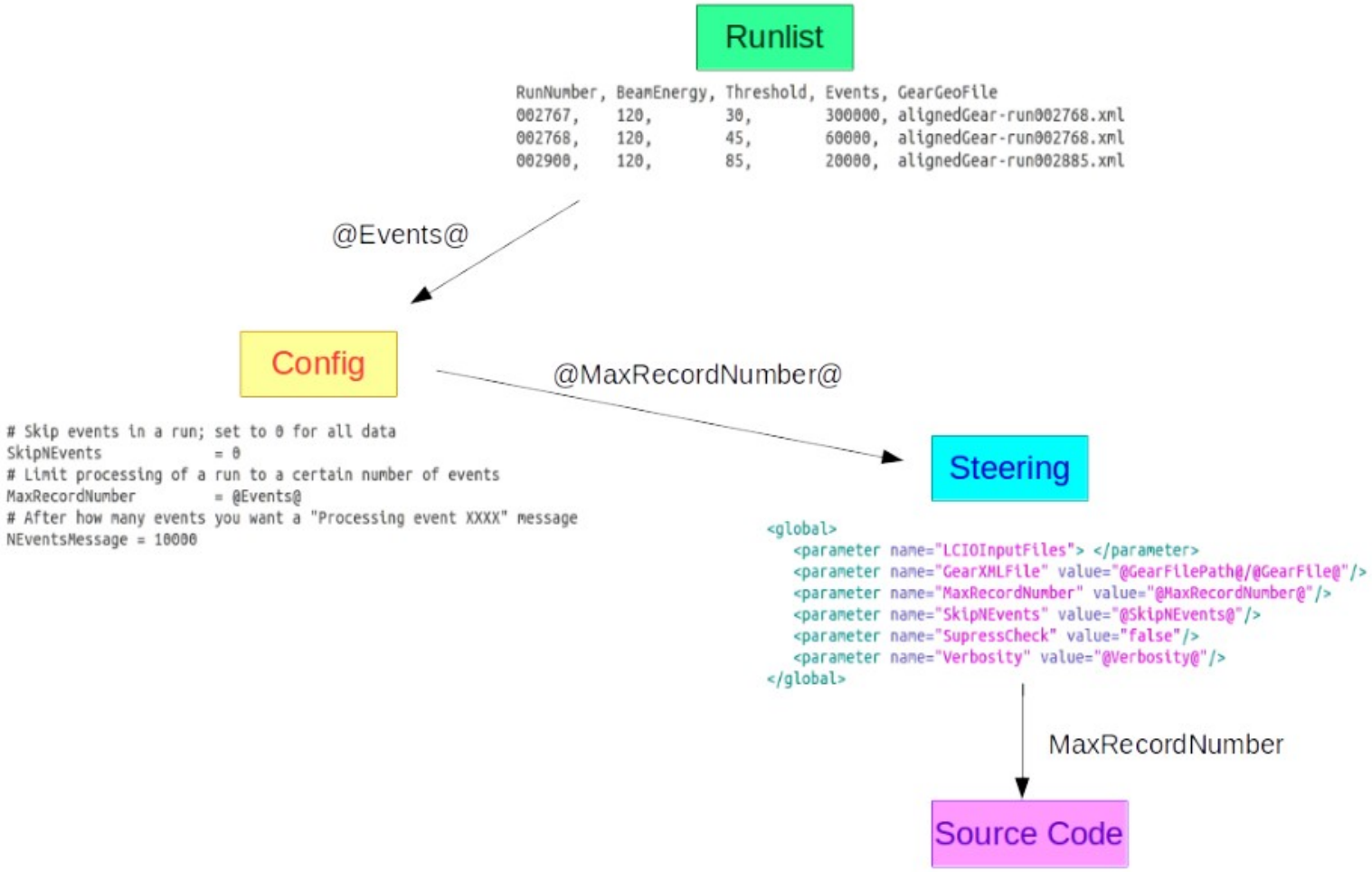
- Written for reconstruction of test beam data
- It is a group of Marlin (Modular Analysis & Reconstruction for the LINear collider) processors
- Embedded in the ILCsoft framework
- Uses the LCIO (Linear Collider I/O) data model
- The geometry is described using the GEAR markup language

Dataset

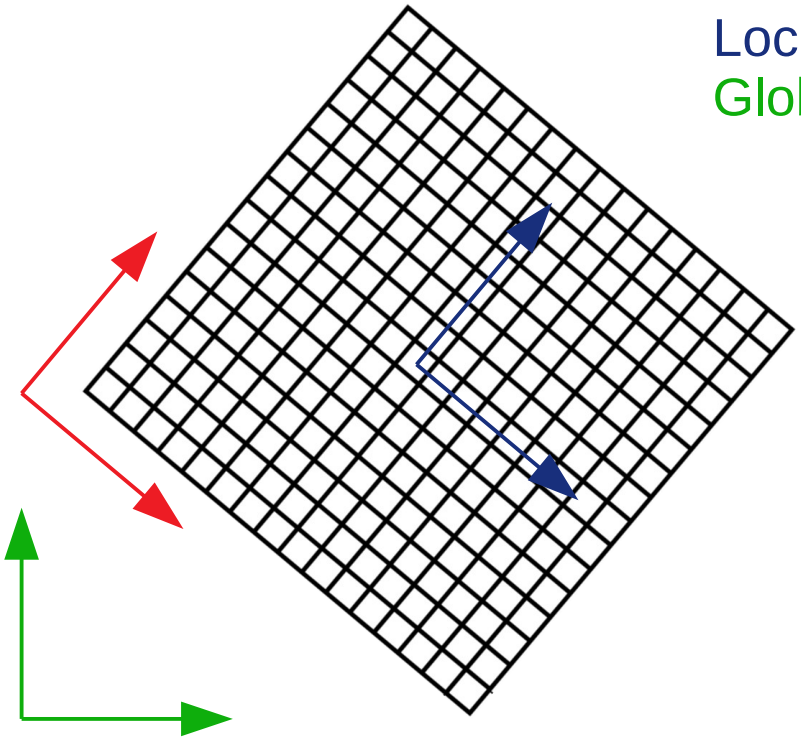
- Six planes EUDET-type telescope (DURANTA)
- 4 GeV electron beam
- ATLAS ITk Strip sensor as active DUT
- ATLAS FEI4 pixel plane as a reference plane
- Raw-to-LCIO conversion performed beforehand with EUDAQ2



EUTelescope Workflow



Coordinate Systems



Pixel coordinate system [pixel numbers]
Local coordinate system [mm]
Global (telescope) coordinate system [mm]

Tutorial Steps

- Setup the EUTelescope environment:

source $EUTELESCOPE/build_env.sh

- Enter in the example folder:

cd $EUTELESCOPE/jobsub/examples/GBL_DUT

- Run the processors:

jobsub -c config.cfg -csv runlist.csv noisypixel 2365

jobsub -c config.cfg -csv runlist.csv clustering 2365

jobsub -c config.cfg -csv runlist.csv hitmaker 2365

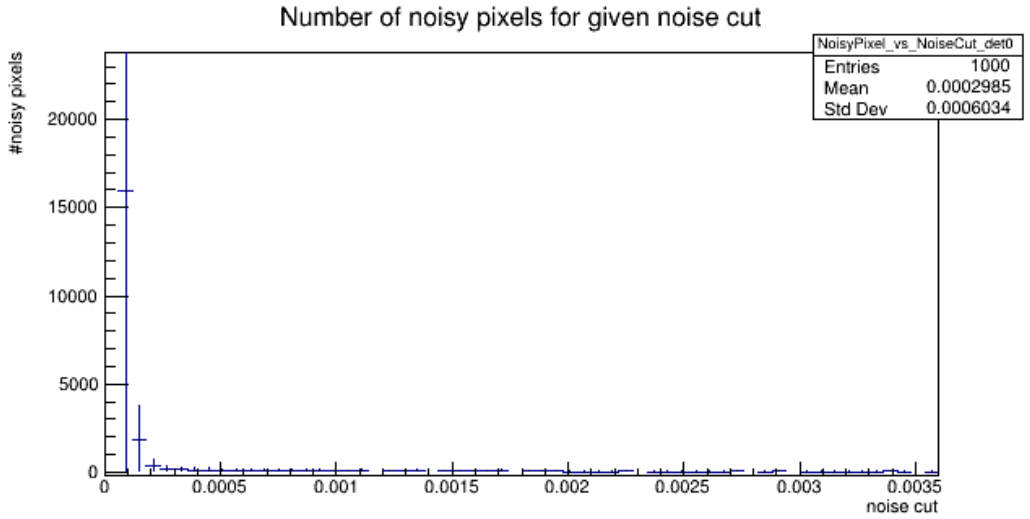
jobsub -c config.cfg -csv runlist.csv alignGBL 2365

jobsub -c config.cfg -csv runlist.csv fitGBL 2365

If you don’t have access to lxplus, you can download the initial file from **https://cernbox.cern.ch/index.php/s/eEztxU06ImNYKOb**

Noisypixel

Previously converter step. With EUDAQ2 the conversion is done externally and this step now is used just to identify (but not mask!) noisy pixels

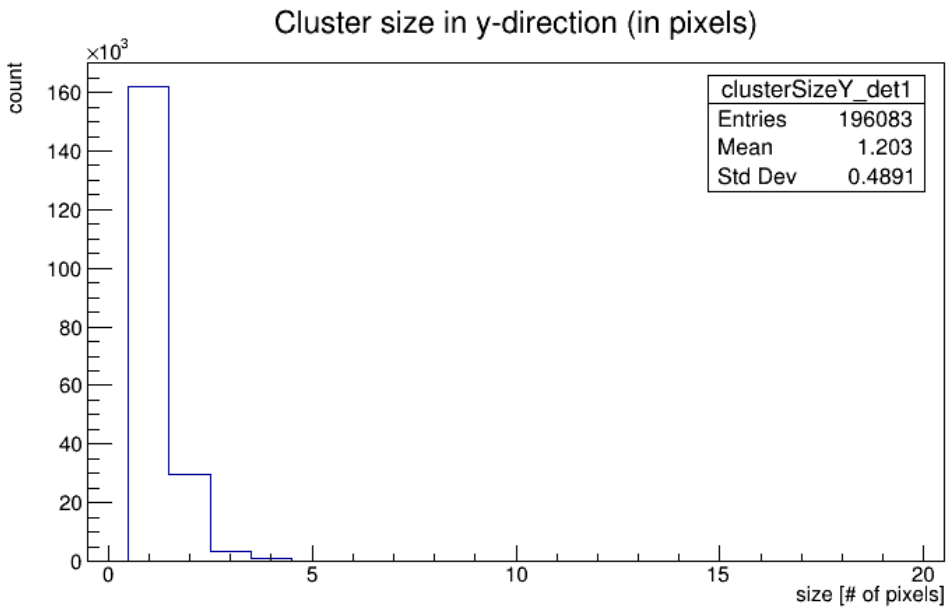


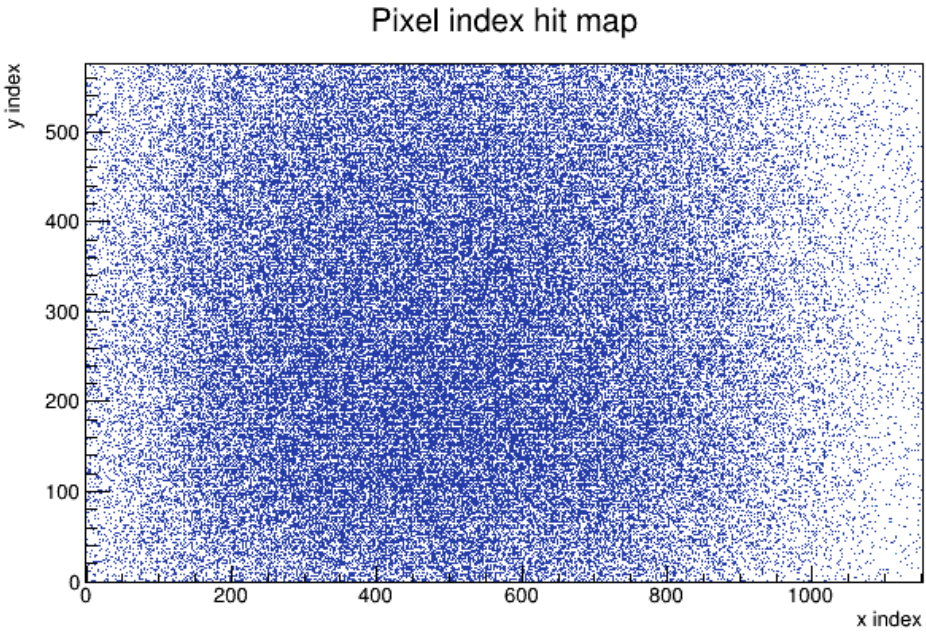
Note that the conversion processor is still present in the steering file, but it it not executed.

```
<execute>
   <processor name="AIDA"/>
   <!-- The raw data has been converter with EUDAQ2 to lcio, thus we don't need to convert it here -->
   <!--processor name="UniversalNativeReader"/-->
   <processor name="NoisyPixelMaskerM26"/>
   <processor name="NoisyPixelMaskerAPIX"/>
   <processor name="Save"/>
   <processor name="EUTelUtilityPrintEventNumber"/>
</execute>
```

Clustering

Groups together single firing pixels to form clusters. Also masks clusters with noisy pixels

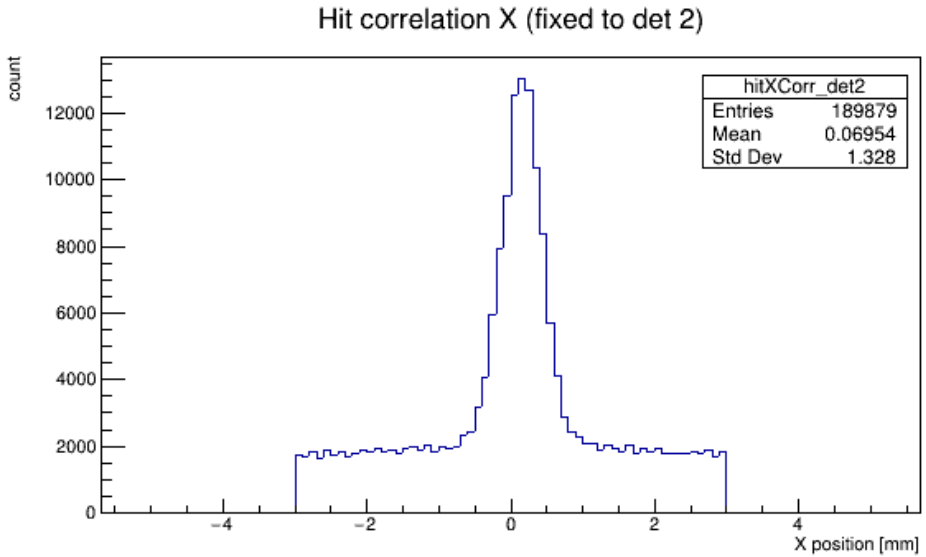


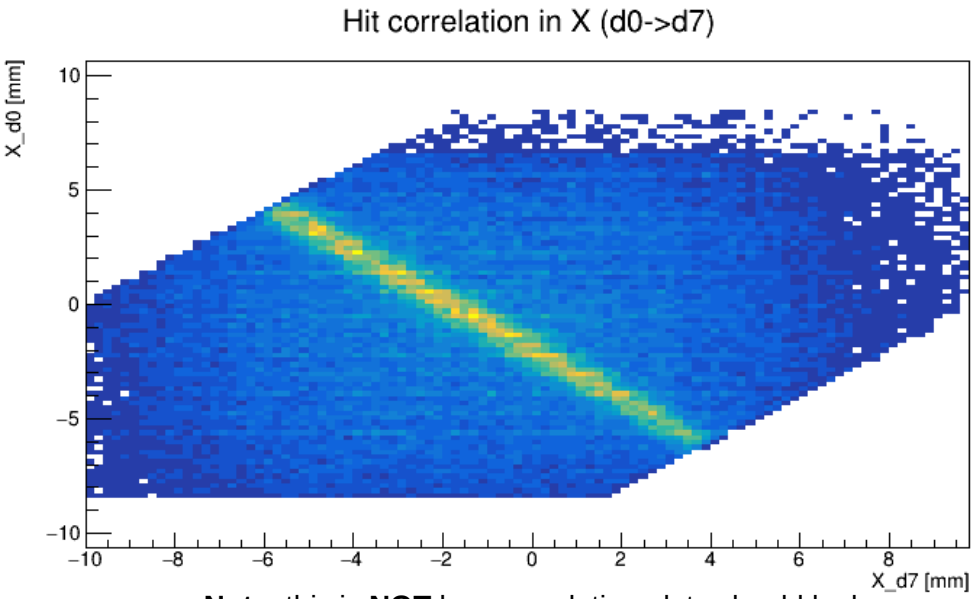


Hitmaker

This step has 3 main purposes:

- Transform the coordinate system from pixel to local (and global) coordinate systems: convert clusters to hits
- Prealignment
- Produce correlation plots for sanity check



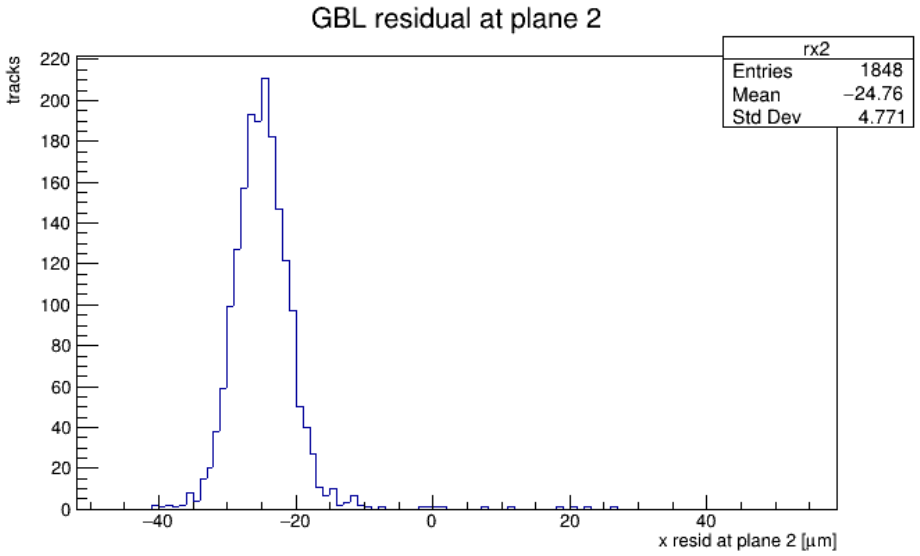


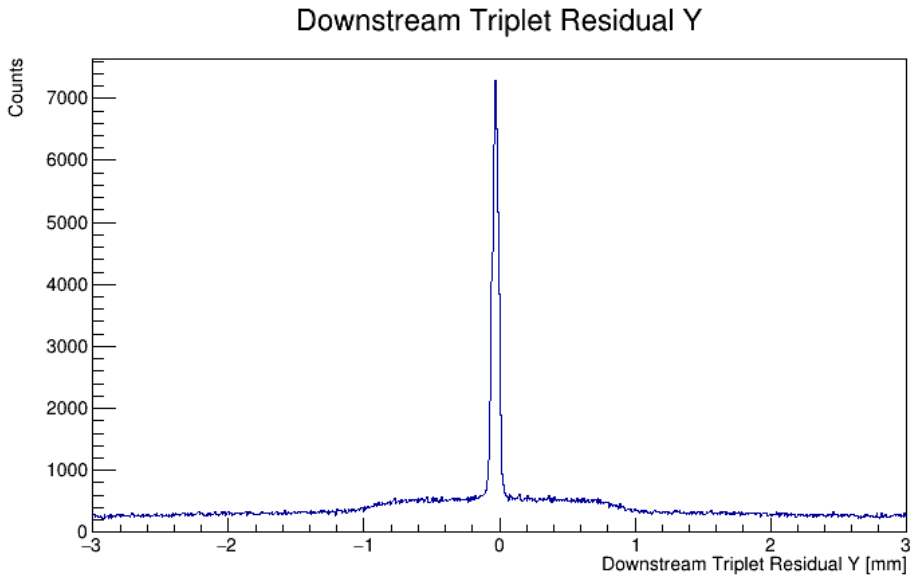
Note: this is **NOT** how correlation plots should look

AlignGBL (1)

Three main steps are executed:

- Track Finding (via triplet method)
- Track Fitting with General Broken Lines (GBL) algorithm
- Geometry alignment through Millepede II

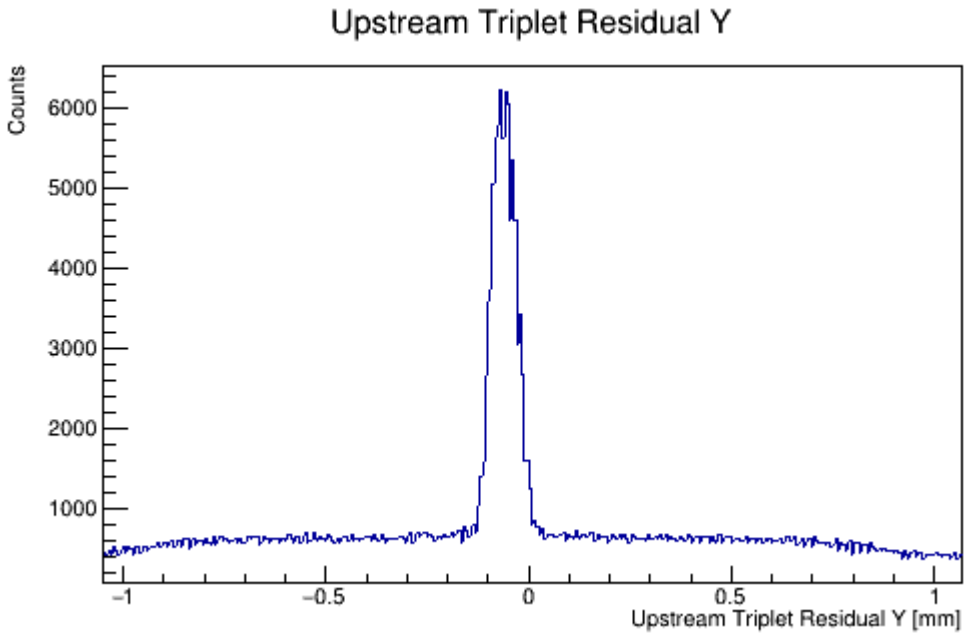




AlignGBL (2)

A few iterations are needed, and at every iterations the cuts for track finding have to be modified accordingly depending on the plots

```
suggestAlignmentCuts = 1
UpstreamSlopeCut = 6
DownstreamSlopeCut = 6
UpstreamTripletResidualCut = 0.03
DownstreamTripletResidualCut = 0.07
TripletMatchingCut = 0.2
DUTCuts = 0.4 1000
```



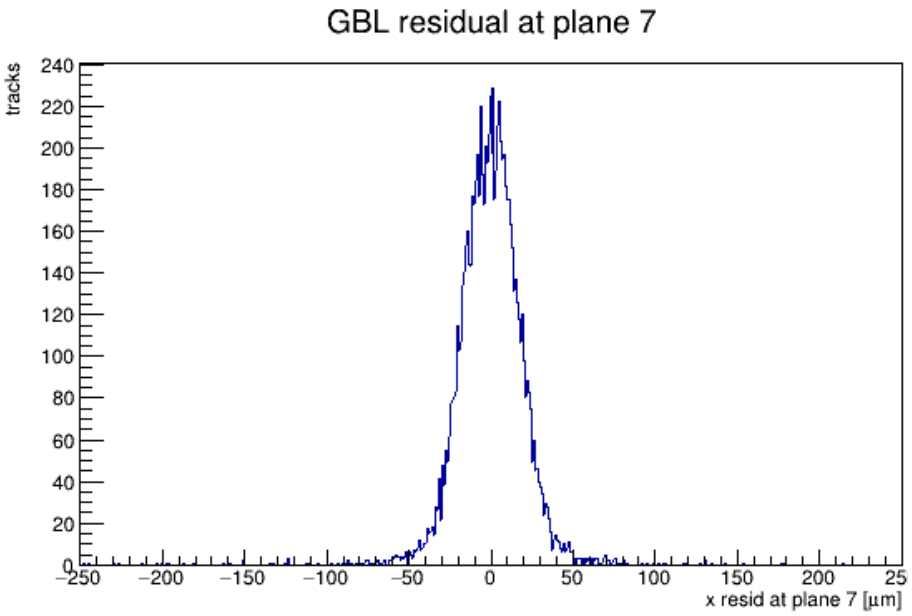
The suggested cuts are printed out in the output. Check them using the plots, though!

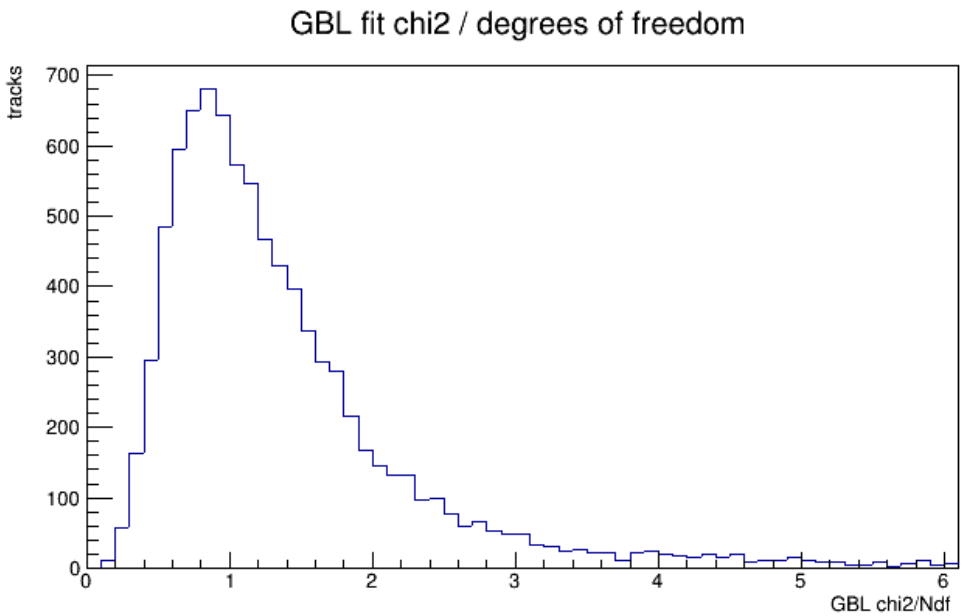
```
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] __________ *** We then recommend the following cuts, although they should be checked ! : __________
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] UpstreamTripletCut     = 0.197356
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] DownstreamTripletCut   = 0.131132
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] UpstreamSlopeCut       = 4.09408
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] DownstreamSlopeCut     = 4.36566
jobsub.alignGBL(INFO): [ MESSAGE5 "MyEUTelGBL"] TripletMatchingCut     = 0.203542
```

fitGBL

Very similar to alignGBL. Again in three steps:

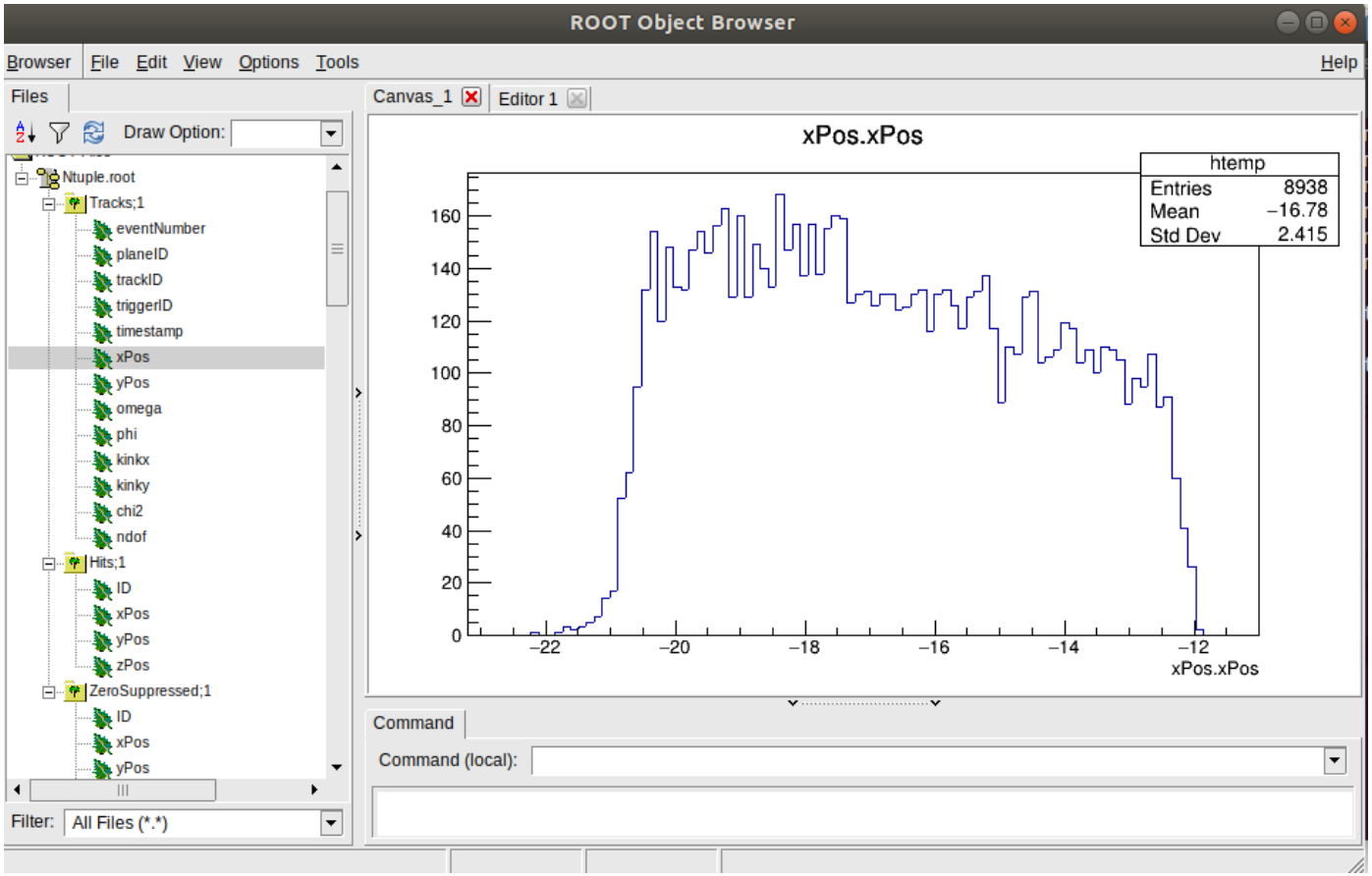
- Track Finding (via triplet method)
- Track Fitting with General Broken Lines (GBL) algorithm
- Track and hit dumping in an NTuple (root file)





Output NTupla

This is the result of the reconstruction and should contain all the information needed to retrieve the information needed by the user (efficiency, resolution, scattering angle, etc...)



(Some) Jobsub Options

- **-c** configuration file (mandatory)
- **-csv** runlist file
- **-o example=1** specifies a variable overriding any definition in the config or runlist files
- **-condor** HTCondor parameter file for batch submission on Condor (NAF or lxplus). An example file can be found in the example condor_submission
- **-lx** Parameter file for batch submission on bsub (lxplus). An example file can be found in legacy/lxplus-submission
- **--dry-run** Generate the steering file for Marlin, without running the processors
- **-g** for colorized output

Some Useful Commands

- Generate a steering file template with all the available processors and parameters:

  Marlin -x > output.xml

- Check a steering file for errors:

  Marlin -u oldsteering.xml newsteering.xml

- Dump information of one event contained in an LCIO file:

  dumpevent file.lcio X

  With X being the number of the event to be checked